What is Claimed is:

1. A software development system for debugging software on a target system having a plurality of processors configured with shared memory, comprising:

a setup utility that receives user input to define a hardware configuration of the target system;

an initialization process that receives the hardware configuration information from the setup utility and creates a software memory map of the target system, loads drivers for each of the plurality of processors, activates a first debug session associated with a first processor of the plurality of processors and activates at least a second debug session associated with a second processor of the plurality of processors wherein each debug session is operable to transmit read requests and write requests to its associated processor; and

a software memory bus that performs processing of shared memory access requests.

2. The software development system of Claim 1 in which the software memory bus performs processing of shared memory access requests using a method for transparently writing to shared memory when debugging a multiple processor system, the method comprising the steps of:

detecting a write request to a shared memory location by the first debug session;

if the first processor associated with the first debug session has write access to the shared memory location

then

selecting the first processor to perform the write request;

else performing the following steps a-b:

a. searching the software memory map to determine if the second processor has write access to the shared memory location;

b. selecting the second processor to perform the write request; and

passing the write request initiated by the first debug session to the selected processor for execution.

3. The method of Claim 2 wherein the step of passing the write request comprises the steps of:

searching the software memory map for a second plurality of processors;

broadcasting the write request to the second plurality of processors; and

performing cache coherency updates in response to the write request in each of the second plurality of processors.

4. The method of Claim 3 wherein the step of broadcasting the write request comprises indicating that the write request is intended for maintaining cache coherency as opposed to a normal write request.

5. The method of Claim 3 wherein the step of performing comprises using cache coherency capabilities, if any, of a processor in response to the write request intended for maintaining cache coherency.

6. The method of Claim 3 wherein:

the step of creating comprises denoting in the software memory map the shared memory locations that contain program instructions;

the step of passing the write request additionally comprises the step of determining that the shared memory location contains a program instruction; and

the cache is an instruction cache.

7. The software development system of Claim 1 in which the software memory bus performs processing of shared memory access requests using a method for maintaining coherency of software breakpoints in shared memory when debugging a multiple processor system, the method comprising the steps of:

setting a first software breakpoint in a shared memory location in the first debug session such that all debug sessions are notified of the setting of the breakpoint; and

clearing the first software breakpoint in the shared memory location in the second debug session such that all debug sessions are notified of the clearing of the breakpoint.

8. The method of Claim 7 wherein the step of setting comprises:

searching the software memory map to find a first plurality of processors having read access to the shared memory location;

updating a software representation maintained for software breakpoints for each of the first plurality of processors; and

writing the software breakpoint instruction in the shared memory location.

9. The method of Claim 7 comprising the step of stepping over a software breakpoint or running after hitting a breakpoint in a shared memory location wherein the method for stepping over the software breakpoint comprises the steps of:

requesting that a software breakpoint in a shared memory location be stepped over or program execution resumed after hitting the breakpoint in a third debug session;

clearing the software breakpoint in the shared memory location in the third debug session such that all debug sessions are notified of the clearing of the breakpoint;

stepping a processor associated with the third debug session to the instruction after the shared memory location from which the software breakpoint was cleared; and

setting the first software breakpoint in the shared memory location in the third debug session such that all debug sessions are notified of the setting of the breakpoint.

10. The method of Claim 9 additionally comprising the steps of

searching the software memory map for a sixth plurality of processors having read access to the shared memory location; and

halting all processors in the sixth plurality of processors after the step of requesting and prior to the step of clearing.

11. The method of Claim 9 wherein the step of setting comprises:

searching the software memory map to find a seventh plurality of processors having read access to the shared memory location;

updating a software representation maintained for software breakpoints for each of the seventh plurality of processors; and

writing the software breakpoint instruction in the shared memory location.

12. The method of Claim 9 wherein the step of clearing comprises:

writing the original instruction stored in a software representation maintained for software breakpoints in the shared memory location;

searching the software memory map to find an eighth plurality of processors having read access to the shared memory location; and

updating a software representation maintained for software breakpoints for each of the eighth plurality of processors to remove the software breakpoint for the shared memory location.

13. The software development system of Claim 1 in which the software memory bus performs processing of shared memory access requests using a method for transparently maintaining cache coherency when debugging a multiple processor system with common shared memory, the method comprising the steps of:

denoting in the software memory map the shared memory locations whether or not each processor of the plurality of processors has a cache;

detecting a write request to a shared memory location by the first debug session;

passing the write request initiated by the first debug session to the first processor for execution;

searching the software representation of the memory map for a first plurality of processors that have read access to the shared memory location;

broadcasting the write request to the first plurality of processors; and

performing cache coherency updates in response to the write request in each of the first plurality of processors.

14. The method of Claim 13 wherein the step of broadcasting the write request comprises indicating that the write request is intended for maintaining cache coherency as opposed to a normal write request.

15. The method of Claim 13 wherein the step of performing comprises using cache coherency capabilities, if any, of a processor in response to the write request intended for maintaining cache coherency.

16. The software development system of Claim 13 in which the software memory bus performs processing of shared memory access requests using a method for transparently maintaining cache coherency when debugging a multiple processor system with common shared instruction memory, the method comprising the steps of:

denoting in the software memory map the shared memory locations whether or not each processor of the plurality of processors has a cache;

detecting a write request to a shared memory location by a first debug session;

if the first processor associated with the first debug session has write access to the shared memory location

then

selecting the first processor to perform the write request;

else performing the following steps a-b:

a. searching the software memory map for a second processor with write access to the shared memory location;

b. selecting the second processor to perform the write request;

passing the write request initiated by the first debug session to the selected processor for execution;

searching the software memory map for a second plurality of processors that have read access to the shared memory location;

broadcasting the write request to the second plurality of processors; and

performing cache coherency updates in response to the write request in each of the second plurality of processors.

17. The method of Claim 16 wherein the step of broadcasting the write request comprises indicating that the write request is intended for maintaining cache coherency as opposed to a normal write request.

18. The method of Claim 16 wherein the step of performing comprises using cache coherency capabilities, if any, of a processor in response to the write request intended for maintaining cache coherency.

19. A digital system, comprising:

multiple processors with common shared memory for executing an application program; and

wherein the application program was developed with the software development system of Claim 1.